ACEEE Int. J. on Signal and Image Processing , Vol. 5, No. 1, January 2014

Efficient Architecture for Variable Block Size Motion Estimation in H.264/AVC

P.Muralidhar, C.B.Rama Rao, and CYN Dwith
Department of Electronics and Communication Engineering, National Institute of Technology-warangal
e-mail: pmurali@nitw.ac.in, cbrr@nitw.ac.in , cyndwith@gmail.com

Abstract - **This paper proposes an efficient VLSI architecture for the implementation of variable block size motion estimation (VBSME). To improve the performance video compression the Variable Block Size Motion Estimation (VBSME) is the critical path. Variable Block Size Motion Estimation feature has been introduced in to the H.264/AVC. This feature induces significant complexities into the design of the H.264/AVC video codec. This paper we compare the existing architectures for VBSME. An efficient architecture to improve the performance of Spiral Search for Variable Size Motion Estimation in H.264/AVC is proposed. Among various architectures available for VBSME spiral search provides hardware friendly data flow with efficient utilization of resources. The proposed implementation is verified using the MATLAB on foreman, coastguard and train sequences. The proposed Adaptive thresholding technique reduces the average number of computations significantly with negligible effect on the video quality. The results are verified using hardware implementation on Xilinx Virtex 4 it was able to achieve real time video coding of 60 fps at 95.56 MHz CLK frequency.**

Keywords: **Motion Estimation, Video compression, Variable Block Size Motion Estimation (VBSME), H.264/AVC, VLSI Architecture.**

I. INTRODUCTION

In many recent standards of video encoders use block matching algorithms (BAMs) for motion estimation because of the efficient and simple implementation. The Motion estimation is a critical component in video coding as it consumes large amount of computational resources. In case of the video coding standard H.264, ME accounts for almost 60% of the complexity of the system. Hence, simplifying the ME process is essential for real-time applications. Several motion estimation algorithms try to reduce the number of search points my employing different motion estimation algorithms [1-3], with loss of video quality. Among these the Three Step Search (TSS), New Three Step Search (NTSS), Four Step Search (4SS), Hexagonal Search (HS) and Diagonal Search (DS) algorithms have been widely accepted and implemented in many video compression standards. These methods reduce the computational complexity of the system, but with considerable degradation in the quality of the video. However, the various search patterns implemented in these algorithms are complex to be implemented on hardware. Hence, the full search block matching algorithms mainly preferred for hardware architectures as they adopted regular computation and achieve high quality of video output. Further, the introduction of VBSME has improved the quality of video coding as fixed-size BMAs have difficulty in accommodating complex movements or motions of small objects within a video frame. Fixed block size might be suitable for low bit rate video coding application, but the high definition video encoding essentially requires efficient methods of motion estimation for better video quality.

The advanced video coding standards such as H.264 [4] lifts this limitation by the employment of VBSME. High coding efficiency is achieved in H.264/AVC through the introduction of multiple reference frames, variable block size motion estimation and other novel algorithms. These features increase the complexity of video codec significantly. In case of VBSME, the current frame is partitioned into non-overlapping macro blocks of required size. In case FSBMA, each MB in current frame the most similar MB in reference frame is obtained using the common criterion of similarity. In H.264/AVC, it supports 7 types of selective block size motion estimation with different sizes of 16x16, 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4 respectively. The computational overhead due to all the candidate modes and the rate distortion optimization calculations essentially increase the complexity. Hence, we need dedicated high-speed and parallel processing architecture implemented through FPGA for real-time encoding of high definition application. The proposed architecture supports all the block size specified in H.264 standard, it also attains low latency and high throughput. The Architecture consists of 16x16 array of Processing Element (PE), each PE can access 4x4 block i.e. 16 pixels, which computes all the 16 SADs in parallel. These SADs of primitive sub blocks are further processed in SAD processor [5], to generate the SADs of other sub blocks. The SAD processer also identifies the best-matching block for each of the 41 sub block concurrently. Thus, the throughput of the architecture is high when compared to other conventional 1D and 2D architectures. The adaptive thresholding algorithms further reduces the com-

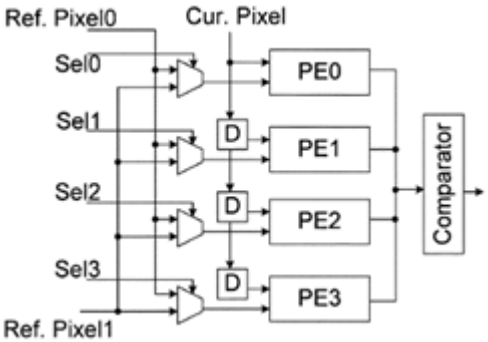

Fig.1 1-D inter-level Hardware Architecture

DOI: 01.IJSIP.5.1.1

putational complexity with negligible effect on the video quality.

The remainder of this paper is organized as follows. In Section II, previous hardware architectures of VBSME are surveyed first, we analyze the impact of supporting VBSME in different hardware architectures. In Section III, we give an Implementation to provide the quantified comparisons of the Spiral Search VBSME with adaptive thresholding. In Section IV, based on our analysis results, we develop hardware architecture for H.264/AVC integer motion estimation. Finally, a conclusion is given in Section V.

II. Motion Estimation Architectures

The Video encoder estimates the motion of the objects between the reference frame and the current frame. This is called motion estimation (ME). It estimates the motion of the pattern corresponding to an object and background in a frame of video sequence. The Block-matching algorithm (BMA) is most widely adopted method for motion estimation in many video compression standards like MPEG 4, H.264/ACV. In case of BMA Motion Estimation, the current frame is divided into a girds of 'Macro Blocks' (MB) then these Macro Blocks are compared with adjacent blocks in the neighborhood of the current frame. Usually the macro block is taken as a square of side 16x16 pixels and the search range of [-16, 16]. The matching of one macro block with another is based on the cost function. The macro block that results in the least cost is the one that matches the closest to current block. Among the various cost function available the most widely used cost function are Sum Absolute Difference (SAD) and Mean Absolute Difference (MAD) given by the equation respectively. SAD is widely used because it is computationally less expensive compared to other methods.

$$SAD = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \left| C_{ij} - R_{ij} \right|$$

$$MAD = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \left| C_{ij} - R_{ij} \right|$$

The H.264/AVC standard is the newest and most efficient video coding standard, reaching compression rates twice higher than the previous standards like MPEG-2. Many features of motion compensation, such as variable block size, multiple reference frames and ¼-pixel accuracy have been included in H.264/AVC. The introduction of VBSME has improved the quality of video coding as fixed-size BMAs have difficulty in accommodating complex movements or motions of small objects within a video frame. Variable block-size motion estimation (VBSME) provides more accurate predictions required for high definition video encoding. In case of VBSME, each macro block constitutes 16 non-overlapping primitive sub blocks. These sub blocks are divided into 7 types of selective block sizes of 16x16, 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4. The smallest of the entire sub block is called the primitive sub block of size 4x4. These primitive blocks are used to derive the other 41 possible sub blocks within a macro block. Hence, for each macro block in VBSME we have 41 motion vectors for 41 different sized sub blocks. Although VBSME can achieve a higher compression ratio, it not only requires huge computation complexity but also increases the difficulty of hardware implementation for ME.

Several architectures have been proposed to reduce the computational overheads in VBSME. In [6], implements H.264 VBSME using the 1D processing elements which reduce the efficiency because of the increase in latency. The architecture contains 16 modules and one VBSME processor, it allows computation of the primitive block SADs in parallel. Such parallel architectures allow the reduction in clock frequency to achieve real-time video coding, frame rate and power dissipation [7]. Another way to support VBSME is to compute the ME for different blocks in parallel by increasing the number of PEs. Reusability of the SADs of primitive blocks to generate SADs of other blocks reduces the computational overhead significantly. These architectures aim at achieving efficient utilization of memory, number of gates, clock frequency and so on. The SAD processing units are the main difference between the FBSME and VBSME in hardware architecture. Hence, the impact of supporting VBSME in hardware architectures is essentially dependent on the different data flows of partial SADs.

A. Work of Yang et al.

Yang et al.[8], was the first to implement the VLSI motion estimation using 1-D inter level hardware architecture as shown in Fig.1. In the given architecture reference pixels are sent to all the PEs and number of PEs is equal to the number of search candidates in the horizontal direction. The control signals are used to send the required reference pixels as inputs to each PE. Registers are used to propagate the current pixels and the partial SAD is stored for each PE. The SADs computed by PE for given reference block are accumulated as shown in Fig.2. The broadcasting technique implemented allows us to reduce the memory bits width i.e. number of bits required for reference data, with the help of some global routing.

However, as compared with the existing 2D architectures, the 1D systolic arrays have longer latency for producing best MVs. Its PE utilization is less than 100 % when $n < N$. Moreover, the 1D architecture cannot search concurrently for the MVs for blocks with different sizes. The throughput of the architecture for VBS-BMA implementation therefore is low.

B. Work of Vos and Stegherr

In [9], Vos and Stegherr proposed a 2-D intra-level architecture where the number of PEs is equal to macro block size. Since, the PE size is equal to the MB size, each PE is associated with a current pixel and this is stored in the respective PEs. The architecture implements a scanning order in the search window which is known as snake scan. In order to realize this, a great deal of propagation registers are used to store reference pixels, and the data in propagation registers

DOI: 01.IJSIP.5.1.1

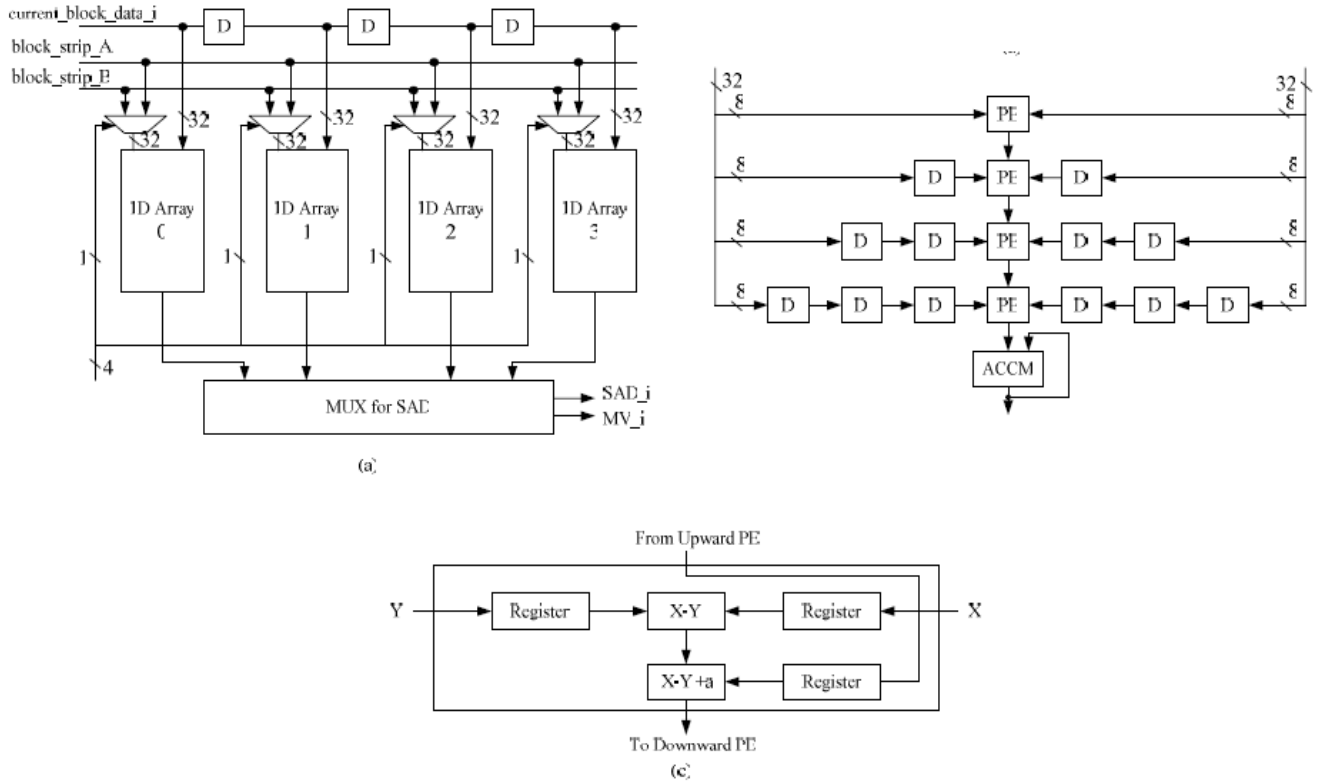

Fig.2 Basic structure of the PE Array (a) structure of the PE array for the module (b) structure of the 1D array in the PE array (c) structure of the PE in 1D array

can be shifted in upward, downward, and right directions. These propagation registers enable us in the reduction of memory usage, but the design essentially contains long latency which limits its performance. In this architecture, each PE computes the partial SAD of the N-rows which are accumulated in the horizontal direction. These partially SADs are further processed to generate the required SAD. In this architecture it does not need to store the partial SADs as the computation of row SADs to SAD is done in one clock cycle.

C. Work of N. Hirai, T. Song, Y. Liu and T. Shimamoto

In [10], proposes an efficient architecture for VBSME using spiral search, the 16 SADs for one search point is calculated in clock cycle using the 16 PE Array of 4x4 modules. The base processing element allows transfer of data in the PE array to top, bottom, right and left. The control signal allows the transfer the data within the PE array to allow spiral search order. The pixel data of current MB is transferred from SRAM to PE Array and saved by a register array in PEs. The current MB is saved until the ME processing for one MB finished. The pixel of reference MB can shift up, down, left or right in each cycle. The SADs of 4x4 blocks are transferred to parallel processing module, 16 SADs of 4x4 PE arrays are used to calculate the SAD of other remaining blocks. These processing is controlled by control unit, and the ME processing is executed.

D. Work of Chien-Min Ou, Chian-Feng Le and Wen-Jyi Hwang

In [5], the architecture consists of 1D array of cascaded to form an PE array as shown in Fig. where the current block size is N=4. It consists of four 1D PE arrays in the architecture and each PE contains 4 PEs. The columns are scheduled through the current primitive sub block using a delay line. Broadcasting is used to set the candidate block of the columns in search region at each clock cycle. Hence, it reduces the latency of PE array when compared to tradition 1D structure. In addition, because of its 100 % PE utilization, the latency of this structure is also lower than that of the conventional 2D systolic architectures.

The synchronous SADs of the primitive blocks computed by the 16 modules allow us to have MV associated with SADs on the same clock cycle. These primitive SADs are used to generate SAD of other subblocks with the help of VBSME processor. In addition to the SAD computation, comparison circuits are included in the 8×8 mode processors and macroblock modeprocessor for identifying the best MV with minimum SAD for each subblock concurrently. Therefore, the best MV for a macroblock can be identified concurrently with the best MVs for all the other 40 subblocks in the macroblock. On the contrary, the conventional 1D or 2D systolic arrays are only able to find the best MV for one block size at a time. The throughput of our architecture therefore is higher than that of the conventional 1D or 2D architectures.

III. Proposed Architecture

The proposed architecture which contains a PE 4x4 array for SAD computation and two local memories for current block and search area as shown in Fig.3. The data of search

DOI: 01.IJSIP.5.1.1

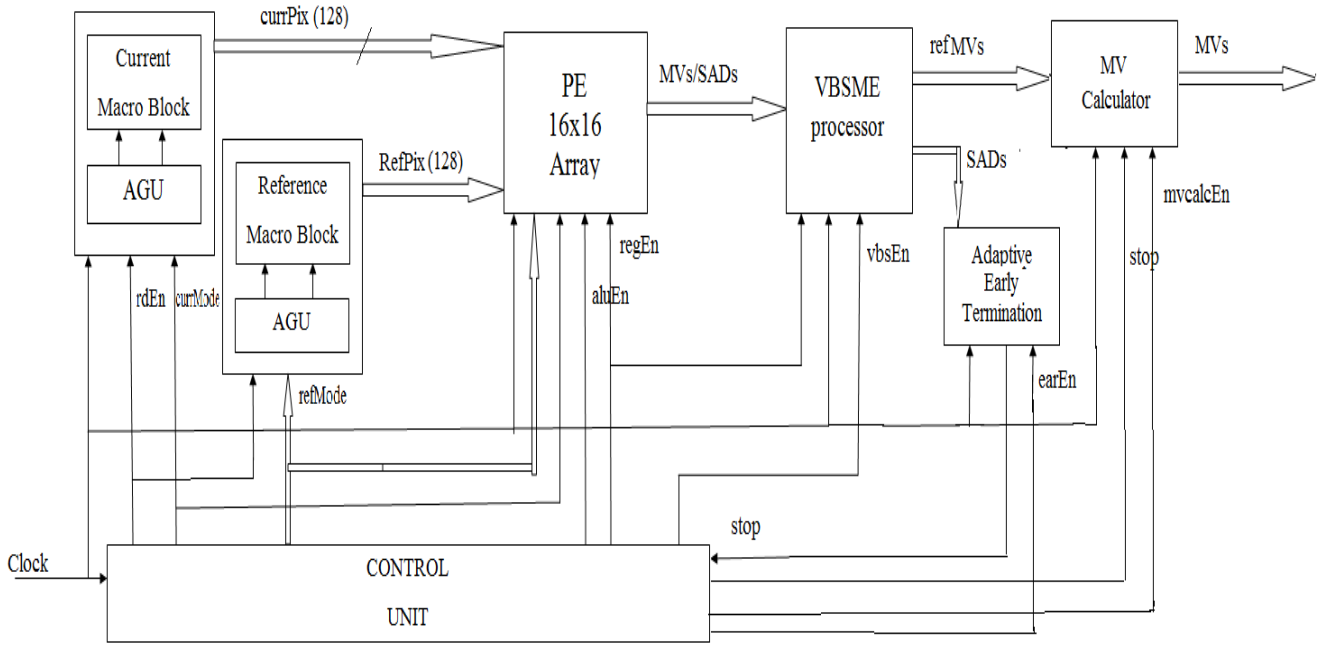

Fig.3 Top Level view of a proposed architecture

areas and current blocks can be inputted into the PE array through the port from the memories for search area and current block. Since the 4x4 block is the smallest block in the MB partitions, 16 SADs of 4x4 blocks in one MB can be computed first through 16 PE 4x4 units and then the SADs of other sub-blocks and MB can be obtained by the 8X8 Mode Calculator(4x8,8x4,8x8) and 16x16 Mode calculator (16x8,8x16,16x16).

A. Processing Element

To support the spiral pattern the modified PE is used such that it can support all four directions as defined in work of Hirai et al [10]. Fig.4 shows the details of a PE, 'Current Block' is an 8-bit register for storing a current pixel. 'Ref Block' is an 8-bit register for storing a candidate pixel. The multiplexers select the input data to load into 'Current Block' or 'Ref Block' registers. The PE module is designed to be able to shift to top, bottom, right or left. The proposed PE can shift reference pixel data to top, bottom, right or left. The current pixel data is always shifting into the PE Array from top direction.

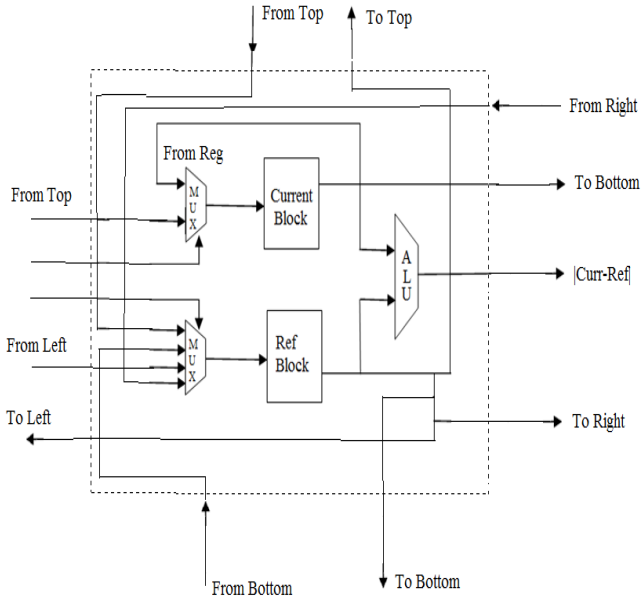

Fig.4 Processing Element

Reference pixel data need input from four directions as well as output to four directions. Therefore, four input ports and four output ports are prepared in each PE. As shown in Fig.5, each PE is connected with surrounding PEs: "from top" connects to "to bottom", "from bottom" connects to "to top", "from left" connects to "to right", and "from right" connects to "to left". An example when "from top" is selected by the multiplexers, the reference data from the output port "to bottom" of upper PEs is shifted to bottom PEs and the search position is shifted. In this way, each I/O port of PE enables the shift of the reference pixel data by selecting the input of reference pixel data using multiplexer. The PE can calculate the difference of every pixel on one clock cycle. It is having two control signals they are 'refpemode' and 'candpemode' these are the modes that tell PE in which direction it has to shift the data stored in PE. Since reference pixel can move in all four directions the 'refpemode' is of 2 bit while for candidate block only one direction is required hence 'candpemode' is of 1 bit.

B. PE 4x4 Array

PE 4x4array calculates the SAD of the primary sub block that is the 4x4 block. There are 16 processing elements (PE) in one PE 4x4 unit can compute synchronously 16 absolute values of difference between the candidate pixels and the current pixels of one 4x4 block. The 4x4 block's SAD can be obtained by accumulating the 16 absolute values of difference in the SAD adder. This structure can calculate the SAD for one search point in one clock cycle using 16 PE Array 4x4

DOI: 01.IJSIP.5.1.1

modules. Moreover, this architecture can make it possible to shift reference pixel data to top, bottom, right or left by connecting PE Array of top, bottom, left or right. Fig. shows the PE4x4 array. In PE4x4 Array the basic element is the processing element so the control signal applied to the processing element is same used here. Since the PE is capable of processing every pixel in a clock cycle and all the PEs are

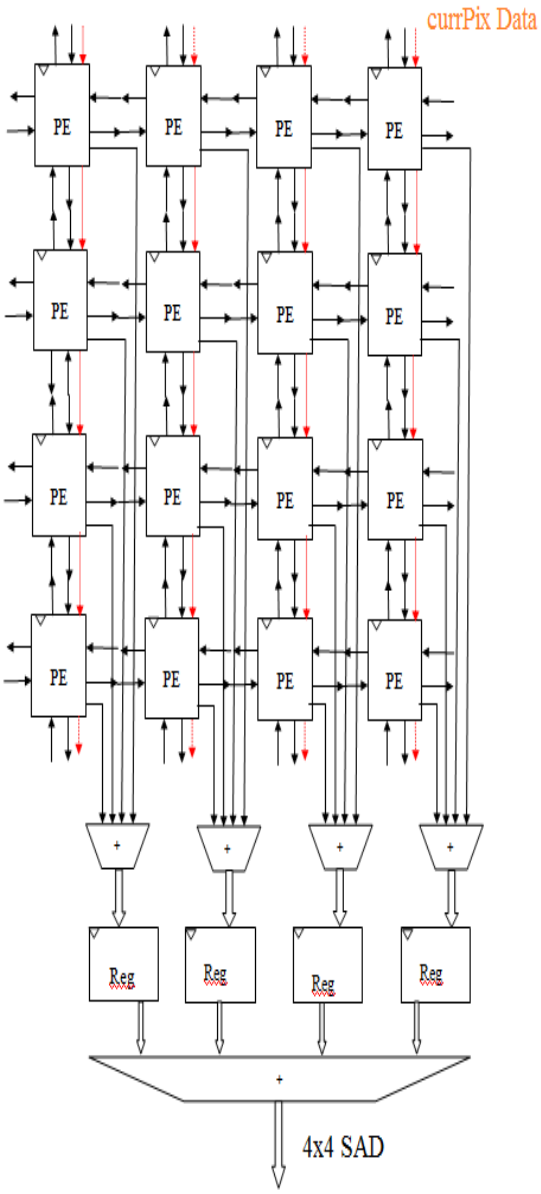

Fig.5 PE 4X4 Array

connected here in parallel hence there all 16 difference will be available simultaneously so registers are employed increase the clock frequency after the difference is calculated all the values are fed to accumulator (SAD calculator) it takes one more extra clock cycle. After difference is calculated by the 4x4 PEs the PE 16x16 array takes one more clock cycle to give the all 16 4x4 SADs. Moreover, this architecture can make it possible to shift reference pixel data to top, bottom, right or left by connecting PE Array of top, bottom, left or right. Fig. shows the PE4x4 array.

C. Variable Block Size Motion Estimation (VBSME) Processor

The SADs of the sub blocks other then 4x4 blocks are calculated by the 8x8 and 16x16 mode calculators using the 4x4 SADs generated from the PE4x4 array. The architecture of these modules has been implemented based on work of chien et al [5]. These calculators comprise of adders which selectively add the 4x4 SADs to generate the higher block SADs. It also computes simultaneously the minimum corresponding SADs and accumulates the minimum SADs. Motion vectors are computed and output when required. The Motion Vector calculator is designed such that it follows the movement of the reference search memory blocks that have been entered into the PE array for calculating the SADs thereby calculating motion vectors of all the 41 block sizes in every clock cycle. Early termination block is connected to the output of the 16x16 mode calculator since the 16x16 SAD is obtained by accumulating all the SADs so if 16x16 SAD is following the threshold condition then most probably all the sub blocks will also follow the condition. So there is no need of applying early termination technique for all sub block thereby reducing the complexity of the early termination block. Once the threshold condition is met the early termination block gives the stop signal thereby stopping the search process for the current block and providing the Motion Vectors corresponding to the reference block. Because all the 16 modules have synchronous SAD computation, the MV

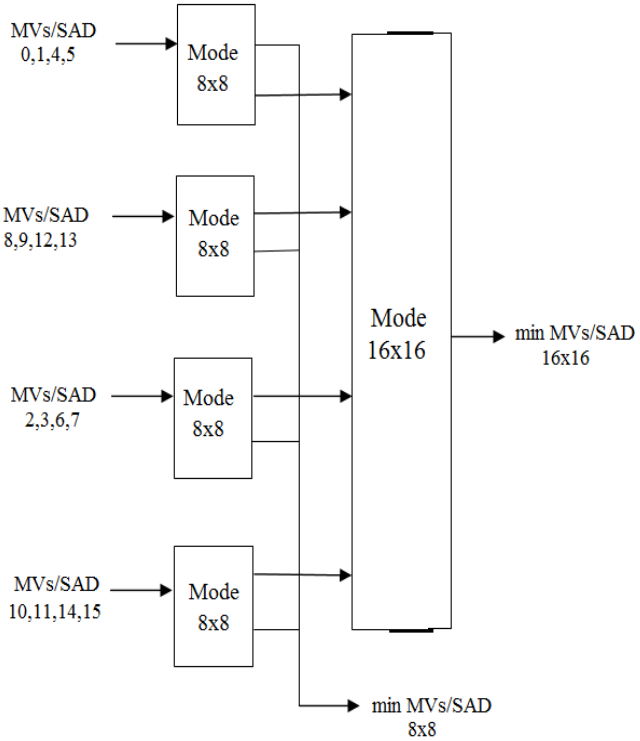

Fig.6 Basic Structure of VBSME processor

associated with SADs produced by these modules are the same on the same clock cycle. The SADs associated with the subblocks of other sizes therefore can be computed by adding the SADs produced by the modules. In our architecture, the VBSME processor, as shown in Fig.6, is used for the SAD

DOI: 01.IJSIP.5.1.1

computation of the subblocks of other sizes.

The VBSME processor contains four 8×8 mode processor, and one macro block mode processor. Each 8×8 mode processor computes the SADs of two 8× 4 sub blocks, two 4× 8 sub blocks, and one 8×8 sub block, as shown in Fig.7.

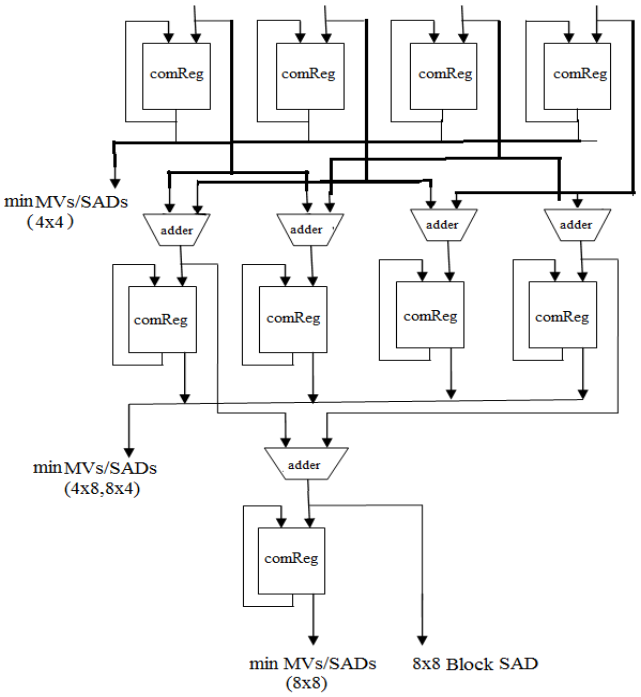

Fig.7 Architecture of 8x8 SAD processor

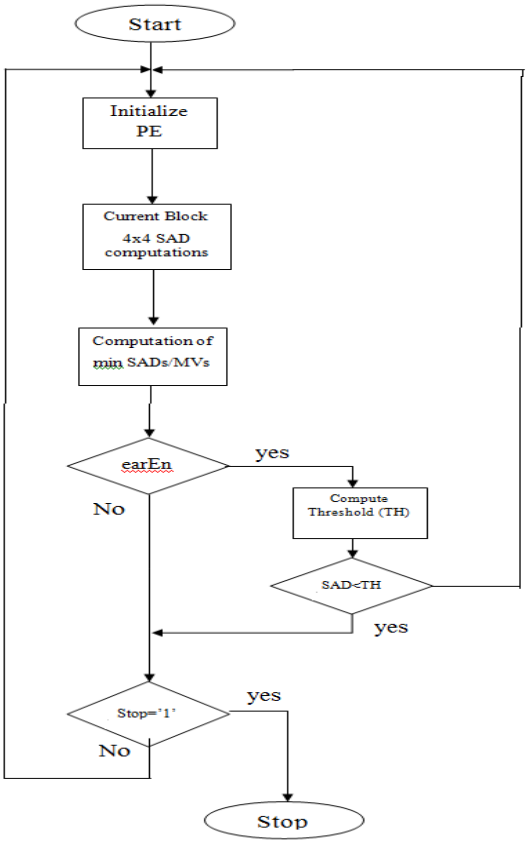

Fig.8 Flow chart of Control Unit

In the macro block mode processor shown in Figure 12, the SAD of four 8×8 sub blocks are used to obtain the SADs of two 16×8 sub blocks, two 8×16 sub blocks, and the 16×16 macro block. In addition to the SAD computation, comparison circuits are included in the 8×8 mode processors and macro block mode processor for identifying the best MV with minimum SAD for each sub block concurrently. Therefore, the best MV for a macro block can be identified concurrently with the best MVs for all the other 40 sub blocks in the macro block. On the contrary, the conventional 1D or 2D systolic arrays are only able to find the best MV for one block size at a time. The throughput of our architecture therefore is higher than that of the conventional 1D or 2D architectures.

For the proposed scheme, an assumption is made that the motion vector of a macro block is most probably related to one of its close neighbours, either in the current frame or in previous frames. The algorithm is designed to find out the optimum threshold to early terminate the search. It is important to note that if the calculated thresholds are too low, then quality restrictions apply meaning that better matches are made, but if the statistics are not accurate, it may take a long time and some macro blocks might even need to use all 961 possible vectors. On the other hand, larger thresholds may lead to more early-exit macro blocks but the quality of the match could be lower. A good solution would require a good initial threshold that will be updated during the macro block's motion search. In addition, spiral search order significantly reduces the memory access cycles. Assume the search range is [-16,16], block-size is 16x16, and 16x1 8 bit pixels data can be loaded in each clock cycle. An extra 15 cycles are required to load the data of the candidate block when the search position is changed to the next line. However, extra cycles can be saved in the adopted scan format because of its ability to scan in all four directions. In this condition, this format can save about 15*(32-1)/ (32*32) H"45% memory access cycles.

D. Control Unit

Controller Unit is used to generate control signals for all the blocks, register enable signals, addresses and enable signals for memory blocks as shown in Fig.8 This is used to generate motion vectors and search point addresses of each reference position during SAD calculation in VBSME processor. Selection lines are generated by controller for selection between current and reference memory data given to the datapath and it is also responsible for final motion vector generation depending on the final address generated from the datapath.

In first state, the current block data and reference block data are loaded into the PE 16x16 array. This state takes 16 clock cycles, each clock cycle load 32 bits of data, 16 form the current Marco Block and 16 from the reference Macro Block. In this state 'rden', 'aguen' signals are enabled. In second state, current block sums are calculated and stored in current register file. As the 4x4 SADs are synchronously calculated in one clock cycle. These SADs are sent to 8x8 mode calculators to generate the 4x8 and 8x4 SADs and MVs. Similarly, the 16x16 mode calculator used to generate 16x16,

DOI: 01.IJSIP.5.1.1

16x8 and 8x16 sub block SADs/MVs simultaneously. In third state, we check for 'earEn' ,which is enable after computation of N(=8) macro block min 16x16 SADs. These SADs are used to generate the required Adaptive Threshold value. If it meets the threshold, then the stop signals is generated which stops further searching else the process continues. There are 272 search points in total and the latency of the circuit is 19 clock cycles. By taking the advantage of high throughput, the circuit is allowed to reduce the clock rate subject to a constraint on frame size and frame rate.

IV. Result And Analysis

MATLAB implementation of the proposed Spiral Search with Adaptive Early Termination algorithms are done on standard test sequences such as Mobile, news, coastguard and Foreman video sequence to validate the implementation of the architecture. The quantitative analysis of the reconstructed image is done using the Peak Signal to Noise Ration given by as shown in Fig.9-11.

$$PSNR = 20\log(\frac{PeakPixelValue}{RootMeanSquareError})$$

Table I: Number of computations in SS and SSET algorithms

Video Sequence	Number of Computations	
	SS	SSET
COAST GUARD	171.94	156.35
FOREMAN	171.94	144.63
MOBILE	171.94	142.23
News	171.94	133.21
HALL	171.94	142.13
Bus	171.94	141.32

Table II: Performance evaluation of SS and SSET

Video Sequence	SS PSNR (dB)	SSAET PSNR(dB)
MOBILE	26.93	26.67
FOREMAN	27.09	26.65
COAST GUARD	26.73	26.23
News	31.57	31.43
Bus	26.16	25.84

Table III: Design Specification

Device Utilized	4vfx100ff1152-12
Algorithm	Spiral Full search with Adaptive Early Termination
# of PEs	16x16 Array
Search range	[-16,16]
Block size	16x16,16x8,8x16,8x8,4x8,8x4 & 4x4
Technology	130nm
Max frequency	95.56 MHz
Critical path	10.465 ns

Latency (T) is the number of clocks required to identify the best MV in all the 41 sub blocks. Throughput (S) of the architecture as the number of best MVs produced per clock cycle. Tables 1-2 present the implementation details of algorithm for various video sequences. Table3 summarizes the design specifications of the system. Table4 shows the latency, throughput and number of PEs, for previous works and proposed architecture. Compared to previous works the proposed architecture takes 272 cycles for processing of each Macro block.

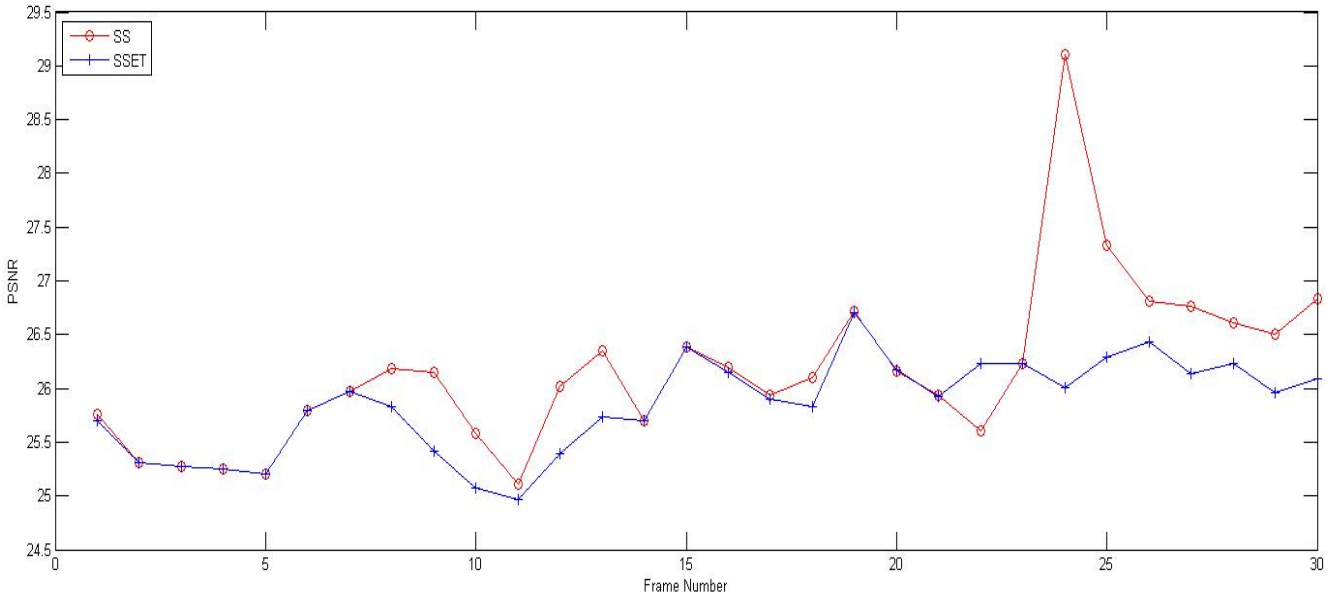

Fig.9 PSNR values of Bus Sequence

DOI: 01.IJSIP.5.1.1

TABLE IV: COMPARISON OF VARIOUS ARCHITECTURES OF BLOCK SIZE (MB=16x16) AND SEARCH RANGE (P= 8)

Architecture	Ref [11]	Ref [12]	Ref [13]	Proposed Arch
No. of PE's	16x16	16x16	16x16	16x16
Latency (L)	5376	4096	305	272
Block Size	16x16, 8x8, 4x4 (Masking)	All Sizes	All Sizes	All Sizes
Throughput (S)	1/256	41/4096	41/305	41/272

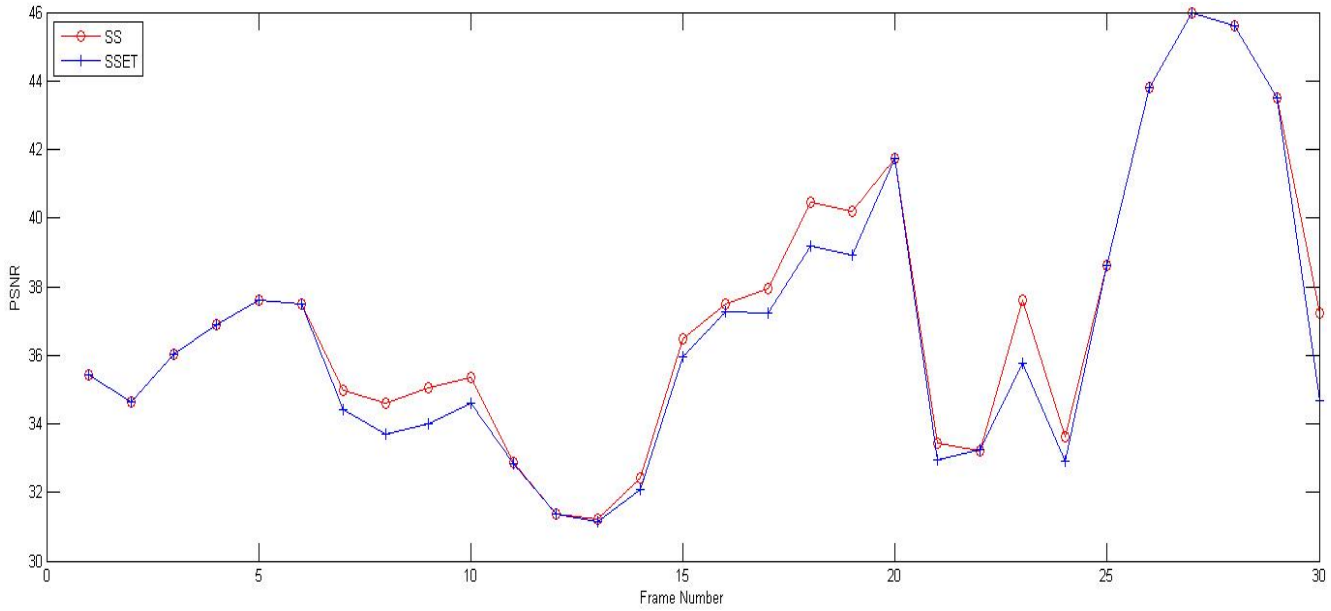

Fig.10 PSNR value of Foreman Sequence

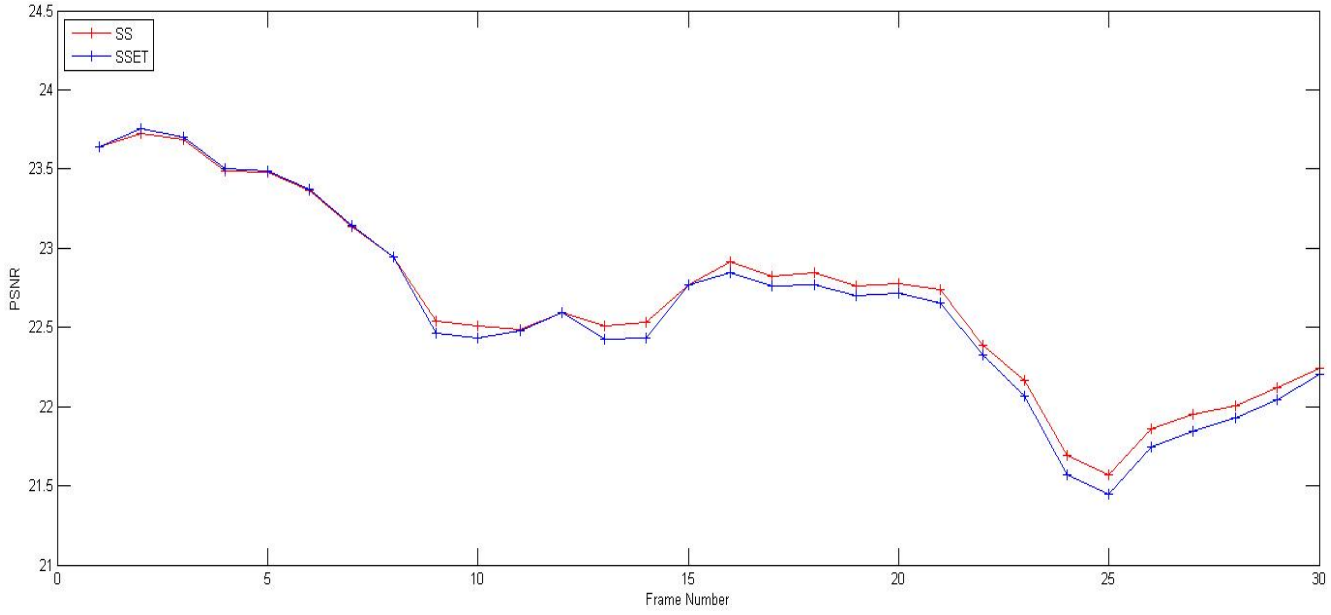

Fig.11 PSNR value of Mobile Sequence

DOI: 01.IJSIP.5.1.1

V. Conclusion

This paper presents an efficient VLSI architecture for VBSME with Spiral Pattern FSBMA in H.264/SVC. The proposed architecture support a "spiral"-like scan format of the search area through a PE array and a memory for the search area. Compared with the scan format with one direction, this format can save about 45% memory access cycles. In our design, 41 MVs of a 16x16 block can be processed in parallel through the reuse of the smaller blocks' SAD. The design can operate at a frequency of 95.56 MHz. Under a frequency of 95.56 MHz, the architecture allows the real-time processing of 1280x720 at 60 fps with FSBMA in a search range [-16, +16]. The proposed architecture provides higher hardware efficiency in terms of gate count and power than previously reported architectures.

References

[1] R. Li, B. Zeng, and M. L. Liuo, "A new three-step search algorithm for block motion estimation," IEEE Transactions on Circuits and Systems for Video Technology, Vol.4, No.4, pp.438.442, Aug. 1994.

[2] S. Zhu, and K.-K. Ma, "A new diamond search algorithm for fast block matching motion estimation," IEEE Transactions on Circuits and Systems for Video Technology, Vol.92, No.2, pp. 287.293, Feb. 2000.

[3] C. Zhu, X. Lin, and L. P. Chau,"Hexagon-based search pattern for fast block motion estimation," IEEE Transactions on Circuits and Systems for Video Technology, Vol.12, No.5, pp.349.355, Aug. 2002.

[4] L. de Vos and M. Schobinger, "VLSI architecture for a flexible block matching processor," IEEE Trans. Circuits and Systems for Video Technology, Vol.5, pp.417-428, 1995.

[5] Chien-Min Ou, Chian-Feng Le and Wen-Jyi Hwang, "An Efficient VLSI Architecture for H.264 Variable Block Size Motion Estimation", IEEE Transactions o 1298 n Consumer Electronics, Vol. 51, No. 4, NOVEMBER 2005.

[6] S.Y. Yap and J.V. McCanny, "A VLSI Architecture for Variable Block Size Video Motion Estimation," IEEE Trans. Circuits and Systems, pp.384-389, Vol. 51, 2004.

[7] A.P. Chandrakasan and R.W. Brodersen, "Minimizing Power Consumption in Digital CMOS Circuits," Proceedings of the IEEE, Vol. 83, pp.498-523, 1995.

[8] K. M. Yang, M. T. Sun, and L. Wu, "A family of VLSI designs for the motion compensation block-matching algorithm," IEEE Trans. Circuits Syst., vol. 36, no. 10, pp. 1317–1325, Oct. 1989.

[9] L. De Vos and M. Stegherr, "Parameterizable VLSI architectures for the full-search block-matching algorithm," IEEE Trans. Circuits Syst., vol. 36, no. 10, pp. 1309–1316, Oct. 1989.

[10] Naoyuki Hirai, Tian Song, Yizhong Liu and Takashi Shimamoto, "A Novel Spiral-Type Motion Estimation Architecture for H.264/AVC", Journal of Semiconductor Technology and Science, Vol. 10, No.1, March, 2010.

[11] L. de Vos and M.Schobinger "A VLSI architecture for flexible block matching processor", IEEE Trans. Circuits and systems for video technology vol 5,pp.417-428,,vol.51,1995.

[12] S.Y.Yap and J.V.McCanny, "A VLSI architecture for variable block size video motion estimation", IEEE Trans. Circuits and systems ,pp384-389, vol 51, 2004.

[13] P.Muralidhar, C.B.Rama Rao and I.Ranjith Kumar, "Efficient Architecture for Variable block size Motion Estimation of H.264 Video Encoder", International Conference on Solid-State and Integrated Circuit (ICSIC), IPCSIT vol. 32, 2012.

DOI: 01.IJSIP.5.1.1